ALC - Project 2: Flying Tourist Problem using SMT

Group 6: Inês Ji 99238, Sebastião Carvalho 99326

20 October 2024

1 Problem description

This project addresses a vacation planning problem for a European tourist who wishes to visit several cities across Europe. The tourist has a fixed number of vacation days and has a defined minimum and maximum number of nights they intend to spend in each city they plan to visit.
The tourist will only travel between cities by plane. The sequence in which these cities are visited is not predefined, only that the trip begins and ends in their home city. The objective is to identify the most cost-effective travel itinerary, ensuring that travel requirements are met: spending a number of nights in each city within the specified minimum and maximum range provided by the tourist, visiting each city exactly once, and starting and returning to the home city. Only direct flights between cities are considered, and no stopovers or layovers are allowed.

2 Install and run

To set up and run the project, follow the steps outlined below:

1. Prerequisites.
Ensure that Python and pip are installed on your system. Detailed instructions for installing Python and pip can be found in this installation guide.

2. Package Installation.
Once Python is installed, install the necessary Python packages by running the following command:

```
pip3 install z3
```

3. Cloning the Repository.
Next, clone the project's GitLab repository to your local machine using the following command:

```
git clone git@gitlab.rnl.tecnico.ulisboa.pt:alc24/project2/6.git
```

4. Running the Project.
After cloning the repository, navigate to the project directory and execute the following command to run the program:

```
./proj2
```

3 Solution

This section presents the solution that was implemented and submitted for the project.

3.1 Domain

Let $\mathbb{F}$ be the set of all possible flights, with cardinality M, and $\mathbb{C}$ be the set of all cities, with cardinality N. The tourist has a total of K vacation days. The set of cities $\mathbb{C}$ is defined as {1,...,N}.

3.2 Variables

The objective is to find the cheapest set of flights, subject to constraints. Considering the domain of the problem, we use flights as our boolean variables. A flight in $\mathbb{F}$ is represented as $f_{i,j,d}$, where i denotes the departure city, j denotes the arrival city, and d is the day of the flight. Additionally, let $p_{i,j,d}$ represent the price of flight $f_{i,j,d}$, and let $k_{j_{min}}$ represent the minimum number of nights to be spent in city j and $k_{j_{max}}$ the maximum number of nights allowed in city j. We consider the BaseCity to be a city $b \in \mathbb{C}$ such that the problem input tells us it is the starting city.

3.3 Encoding

3.3.1 Hard clauses

1. Depart from each city only once

Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{c,i,d} = 1$$

Explanation: The tourist must depart from each city exactly once, therefore only one flight departing from each city can be taken.

2. Arrive at each city only once

Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{i,c,d} = 1$$

Explanation: The tourist must arrive at each city exactly once, therefore only one flight arriving at each city can be taken.

3. Stay between $k_{j_{min}}$ and $k_{j_{max}}$ nights in city j

Formal description:

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee \neg f_{j,h,e}, \quad \text{if } e > d + k_{j_{max}} \vee e < d + k_{j_{min}}$$

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, \neg f_{i,j,d} \vee (\sum f_{j,h,e} \geq 1), \quad else$$

Explanation: The tourist must stay in each city j for a duration that is in between $k_{j_{min}}$ and $k_{j_{max}}$ nights. This means that if the tourist arrives in city j on day d, they must depart from that city on a flight scheduled on a day between $d + k_{j_{min}}$ and $d + k_{j_{max}}$. Consequently, any flights departing from city j before $d + k_{j_{min}}$ nights or after more than $d + k_{j_{max}}$ nights have passed are not valid options. The tourist needs to take a flight from city j in this time frame.

4. End in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, k \in \mathbb{C}, \neg f_{i,b,d} \vee \neg f_{j,k,e}, \quad \text{if } (e \geq d), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

Explanation: Once the tourist flies to the base city, no further flights can be taken. The last flight must be to the base city.

5. Start in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, k \in \mathbb{C}, \neg f_{b,i,d} \vee \neg f_{j,k,e}, \quad \text{if } (e \leq d), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

Explanation: The first flight of the trip must be from the base city. No flights that occur before this one can be taken.

6. Remove flights with invalid departure date

Formal description:

$\mathbb{F}^{i}_{d,e}$ is the set of all flights in $\mathbb{F}$ that arrive at city i between days d and e (including d and e).

$$\forall d \in \{1, .., K\}, \forall j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d}, \quad \text{if } \mathbb{F}^{i}_{max(1,d-k_{i_{max}}),d-k_{i_{min}}} = \emptyset \quad \text{with } (d > k_{i_{min}})$$

$$\forall d \in \{1, .., K\}, \forall j \in \mathbb{C}, \forall i \in (\mathbb{C} \setminus \text{BaseCity}), \neg f_{i,j,d}, \quad \text{if } (d \leq k_{i_{min}})$$

Explanation: A flight $f_{i,j,d}$ (excluding those departing from the base city) is not valid if there is no flight arriving in city i from any city c on day e, such that $d - e$ is within the range between $k_{i_{min}}$ and $k_{i_{max}}$.

7. Remove invalid base city flights

Formal description:

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, \neg f_{i,b,d}, \quad \text{if } (d < \sum_{j=1}^{N} k_{j_{min}}), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, \neg f_{b,i,d}, \quad \text{if } (d > K - (\sum_{j=1}^{N} k_{j_{min}})), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

Explanation: The tourist cannot arrive back at the base city before spending the minimum of nights in each city. Therefore, flights arriving at the base city before the sum of all minimum nights have passed are invalid. Similarly, the tourist cannot depart from the base city when fewer than the sum of all minimum nights remain, as it would be impossible to visit all planned cities. As a result, any flight departing from the base city with fewer than the sum of all minimum nights remaining is not valid.

3.3.2 Soft clauses

Flights as soft clauses
Formal description:

$$\forall f_{i,j,d} \in \mathbb{F}, \neg f_{i,j,d}, \text{with } weight = p_{i,j,d}$$

Explanation: For each flight, its negation is added as a soft clause with its price as the weight. The goal is to minimize the total cost of the flights taken while satisfying the hard constraints.

4 Alternative solution

This section presents an alternative approach that was explored during the project.

4.1 Domain

Let $\mathbb{F}$ be the set of all possible flights, with cardinality M, and $\mathbb{C}$ be the set of all cities, with cardinality N. The tourist has a total of K vacation days. The set of cities $\mathbb{C}$ is defined as {1,...,N}.

4.2 Variables

The objective is to find the cheapest set of flights, subject to constraints. Considering the domain of the problem, we use flights as our boolean variables. A flight in $\mathbb{F}$ is represented as $f_{i,j,d}$, where i denotes the departure city, j denotes the arrival city, and d is the day of the flight. We also introduce an integer variable c_j, which represents the number of nights the tourist stays in city j. Additionally, let $p_{i,j,d}$ represent the price of flight $f_{i,j,d}$, and let k_{jmin} represent the minimum number of nights to be spent in city j and k_{jmax} the maximum number of nights allowed in city j. We consider the BaseCity to be a city $b \in \mathbb{C}$ such that the problem input tells us it is the starting city.

4.3 Encoding

4.3.1 Hard clauses

1. Depart from each city only once
Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{c,i,d} = 1$$

Explanation: The tourist must depart from each city exactly once, therefore only one flight departing from each city can be taken.

2. Arrive at each city only once
Formal description:

$$\forall c \in \mathbb{C}, \sum_{d=1}^{K} \sum_{i=1}^{N} f_{i,c,d} = 1$$

Explanation: The tourist must arrive at each city exactly once, therefore only one flight arriving at each city can be taken.

3. Stay c_j nights in city j
Formal description:

$$\forall j \in (\mathbb{C} \setminus \text{BaseCity}), \forall i, h \in \mathbb{C}, \forall d, e \in \{1, ..., K\}, (e - d \neq c_j) \implies ((\neg f_{i,j,d}) \vee (\neg f_{j,h,e}))$$

Explanation: The tourist must stay in each city j for exactly c_j nights. This means that if the tourist arrives in city j on day d, they must depart from city j on day $d + c_j$ to stay exactly c_j nights. Therefore, if there are no flights departing from city j on day $d + c_j$, then either the departure flight to city j or the arrival flight from city j is not a valid option.

4. End in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, h \in \mathbb{C}, (e \geq d) \implies ((\neg f_{i,b,d}) \vee (\neg f_{j,h,e})), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

Explanation: Once the tourist flies to the base city, no further flights can be taken. The last flight must be to the base city.

5. Start in base city

Formal description:

$$\forall d, e \in \{1, .., K\}, \forall i, j, h \in \mathbb{C}, (e \leq d) \implies ((\neg f_{b,i,d}) \vee (\neg f_{j,h,e})), \quad \text{with } b \in \mathbb{C} \text{ being the base city}$$

Explanation: The first flight of the trip must be from the base city. No flights that occur before this one can be taken.

6. Remove flights with invalid departure date

Formal description:

$\mathbb{F}^j$ is the set of all flights $f_{i,j,d}$ that arrive at city j.

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, \forall j \in C, (\bigwedge_{f_{k,i,e} \in \mathbb{F}^i} (d - e \neq c_i)) \implies (\neg f_{i,j,d})$$

Explanation: A flight $f_{i,j,d}$ (excluding those departing from the base city) is not valid if there is no flight arriving in city i from any city c on day e, such that $d - e$ is equal to c_i.

7. Remove invalid base city flights

Formal description:

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, (d < (\sum_{j=1}^{N} c_j)) \implies (\neg f_{i,b,d}) \quad b \text{ is the base city}$$

$$\forall d \in \{1, ..., K\}, \forall i \in \mathbb{C}, (d + (\sum_{j=1}^{N} c_j) > K) \implies (\neg f_{b,i,d}), \quad b \text{ is the base city}$$

Explanation: The tourist cannot arrive back at the base city before spending c number of nights in each city. Therefore, flights arriving at the base city before the sum of all nights of all cities have passed are invalid. Similarly, the tourist cannot depart from the base city when fewer than the sum of all nights of all cities remain, as it would be impossible to visit all planned cities. As a result, any flight departing from the base city with fewer than the sum of all nights remaining is not valid.

8. City nights constraint

Formal description:

$$\forall j \in \mathbb{C}, (k_{j_{min}} <= c_j <= k_{j_{max}})$$

Explanation: The number of nights the tourist stays in city j must be between the minimum ($k_{j_{min}}$) and maximum ($k_{j_{max}}$) limits specified for that city.

4.3.2 Soft clauses

Flights as soft clauses

Formal description:

$$\forall f_{i,j,d} \in \mathbb{F}, \neg f_{i,j,d}, \text{with } weight = p_{i,j,d}$$

Explanation: For each flight, its negation is added as a soft clause with its price as the weight. The goal is to minimize the total cost of the flights taken while satisfying the hard constraints.so

5 Algorithm

To solve the problem, we employ the Maximum Satisfiability (MaxSat) for SMT approach, utilizing the Z3 Optimizer solver with default flags.

6 Encoding comparison

In this section, we compare two encoding approaches we tried in order to determine which encoding performs better under different conditions.

6.1 Differences between the encodings

The first solution uses only boolean encoding, where the variables represent the flights. The second solution extends the first by introducing integer variables to represent the number of nights the tourist stays in each city.
While the first solution has fewer variables, it introduces broader conditions when applying constraints. This is because it has to check flight validity across a range of possible days (i.e., within an interval) to ensure compliance with the constraints. The second solution, by contrast, uses integer variables to represent the exact number of nights in each city, This allows for more precise constraint checking since it compares a specific value (the number of nights) rather than an interval.

6.2 Experimental results & analysis

The performance of the two solutions was compared using the set of public instances, and the results were measured in terms of execution time and memory usage of the solvers. Both encodings were run on the same machine with the same test inputs, which varied in the number of cities and flights.

Tests	Number of cities	Number of flights	1st Solution time (s)	2nd Solution time (s)
t01	3	22	0	0
t02	4	25	0	0
t03	2	10	0	0
t04	3	53	0	0
t05	3	44	0	0
t06	4	109	0	1
t07	4	198	1	2
t08	5	221	1	2
t09	5	319	2	3
t10	6	456	3	8
t11	6	530	4	10
t12	7	1086	38	88
t13	6	774	10	28
t14	7	838	14	37
t15	8	1176	88	121
t16	8	1181	73	182
t17	8	776	12	35
t18	8	882	14	46

Table 1: Execution time comparison between two solutions

Tests	Number of cities	Number of flights	1st Solver Memory usage (MB)	2nd Solver Memory usage (MB)
t01	3	22	17.49	17.72
t02	4	25	17.59	17.74
t03	2	10	17.47	17.58
t04	3	53	17.66	18.22
t05	3	44	17.66	18.14
t06	4	109	18.25	20.18
t07	4	198	19.21	24.15
t08	5	221	19.99	24.84
t09	5	319	22.63	40.52
t10	6	456	28.57	72.33
t11	6	530	38.33	78.06
t12	7	1086	143.13	324.72
t13	6	774	69.54	226.13
t14	7	838	88.18	256.02
t15	8	1176	156.21	314.52
t16	8	1181	156.98	455.92
t17	8	776	77.99	235.28
t18	8	882	78.89	258.92

Table 2: Memory usage comparison of the solvers between two solutions

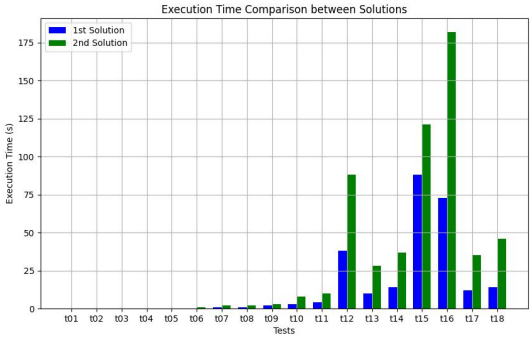

Figure 1: Execution time across 18 test cases

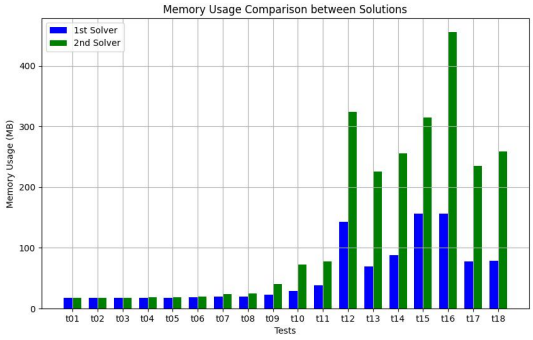

Figure 2: Memory usage across 18 test cases

Table 1 and **Figure 1** shows the execution time for both encodings across 18 test cases. As the complexity of the test cases increases, with more cities and flights, the second solution generally takes longer due to the additional complexity of using integer variables.
Table 2 and **Figure 2** shows the memory usage of the solvers of the two solutions across 18 test cases. As expected, the first solution is more memory-efficient due to the absence of integer variables. The second solution consumes significantly more memory, especially as the number of cities and flights increases, because of the additional complexity introduced by the integer variables representing the number of nights.

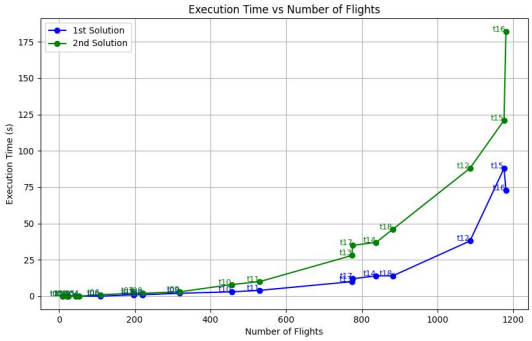

Figure 3: Flights per execution time

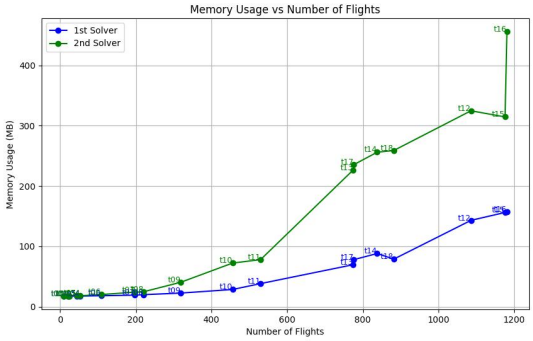

Figure 4: Flights per memory usage

Figure 3 and **Figure 4** highlight how the number of flights influence the execution time and memory usage, respectively, for both solvers. Both charts display an exponential curve since as the number of flights increases, the second solution's execution time and memory usage grow at a significantly higher rate compared to the first solution due to its additional complexity introduced by integer variables. For lower flight counts both solutions show minimal differences.

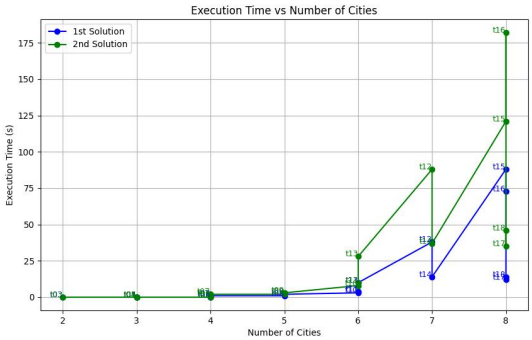

Figure 5: Cities per execution time

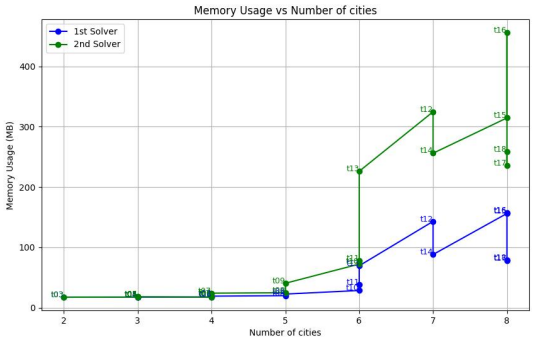

Figure 6: Cities per memory usage

Figure 5 and **Figure 6** display how the number of cities influences the execution time and memory usage, respectively, for both solvers. Although we observe an increase in both time and memory usage as the number of cities rises, this is largely correlated with the increase in the number of flights. For example, in cases *t13* and *t17*, where the number of flights is nearly the same (774 vs. 776), but the number of cities rises from 6 to 8, we do observe an increase in both time and memory usage, though it is not as drastic as when the number of flights increases. A broader range of number of cities would be needed to analyze the isolated impact of cities more effectively.

6.3 Conclusions

From the results, it is evident that the first solution, which uses only boolean variables, performs significantly better in terms of both execution time and memory usage compared to the second solution, which introduces integer variables. As the complexity of the problem increases, particularly with more cities and flights, both solutions show exponential growth in execution time and memory usage. However, the second solution's resource demands grow at a much faster rate, due to the additional complexity introduced by integer variables. For smaller problem instances, this difference is negligible, but as the number of cities and flights rises, the advantage of the first solution's simplicity becomes more apparent.
In summary, the first solution, which uses boolean variables only, is better suited for larger instances where memory and time efficiency are critical unlike the second solution that while providing more precise constraint checking, incurs a significant performance cost as the complexity increases.